Security at the Network Layer: IPSec

Sachin Tripathi

IIT(ISM), Dhanbad

IP Security (IPSec)

It is a collection of protocols designed by the Internet Engineering Task Force (IETF) to provide security for a packet at the network level. The network layer in the Internet is often referred to as the Internet Protocol or IP layer. IPSec helps create authenticated and confidential packets for the IP layer

IPSec can be useful in several areas.

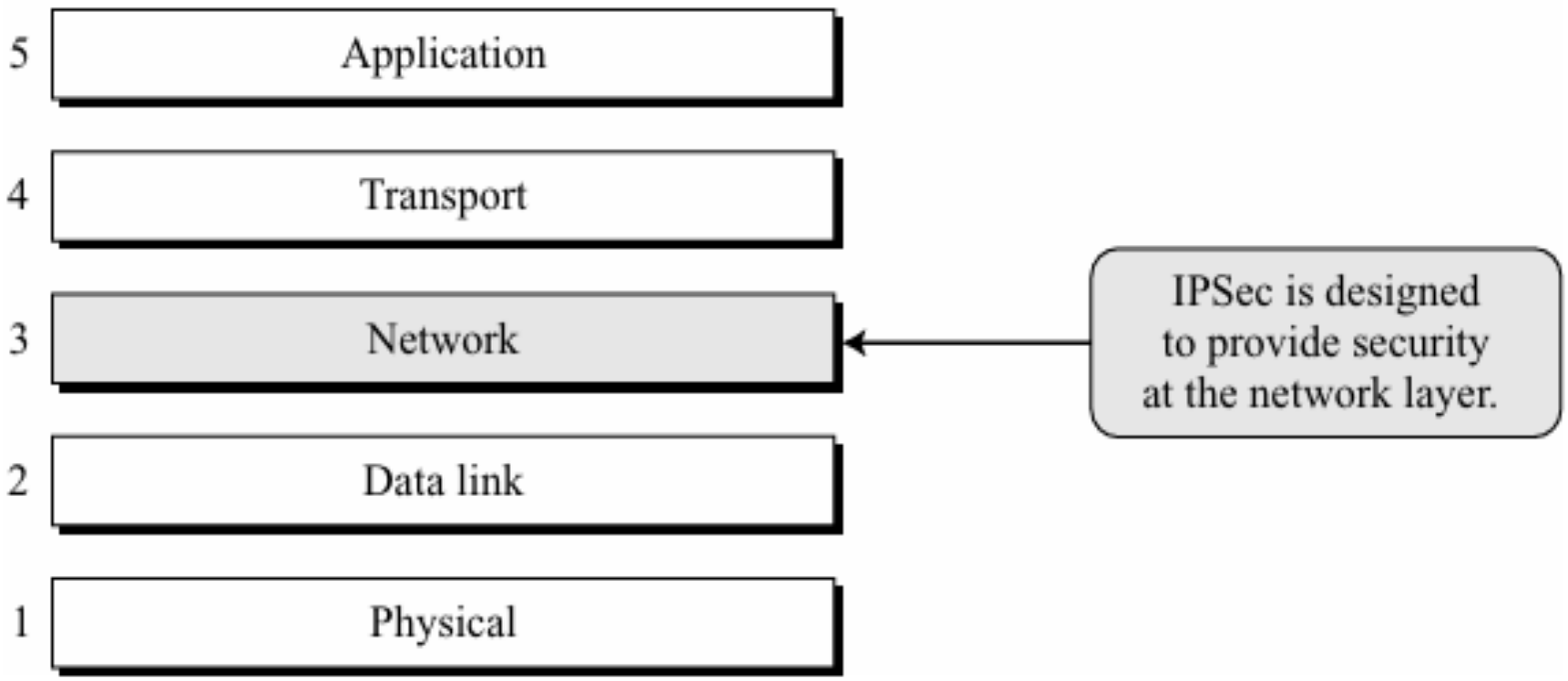
First, it can enhance the security of those client/server programs, such as electronic mail, that use their own security protocols.

Second, it can enhance the security of those client/server programs, such as HTTP, that use the security services provided at the transport layer.

It can provide security for those client/server programs that do not use the security services provided at the transport layer.

It can provide security for node-to-node communication programs such as routing protocols.

TCP/IP protocol suite and IPSec



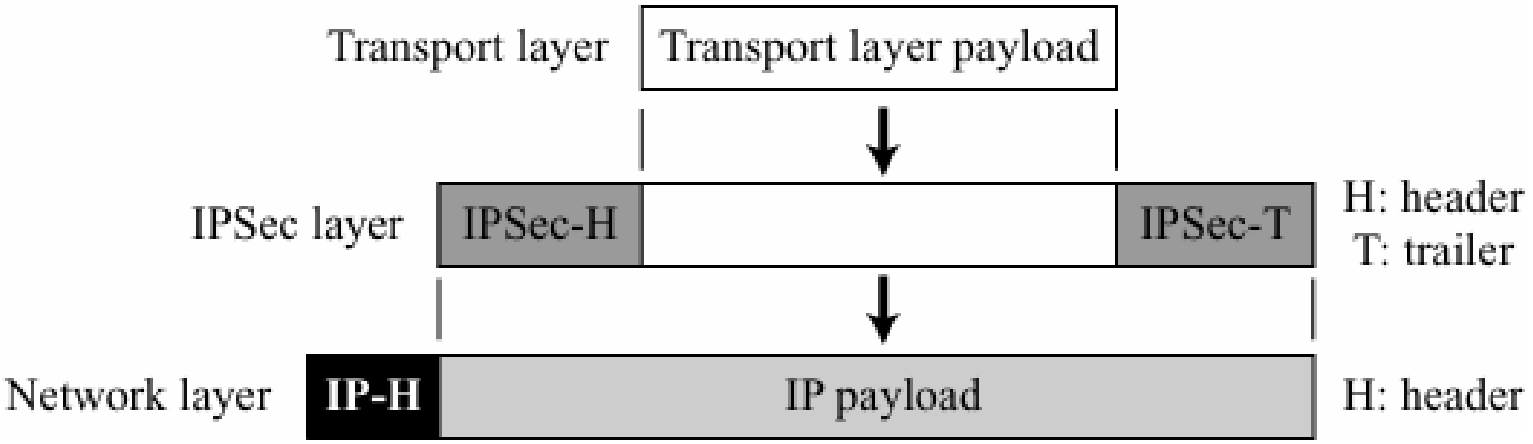
TWO MODES (Transport & Tunnel)

IPSec operates in one of two different modes: **transport mode or tunnel mode.**

- **Transport Mode**

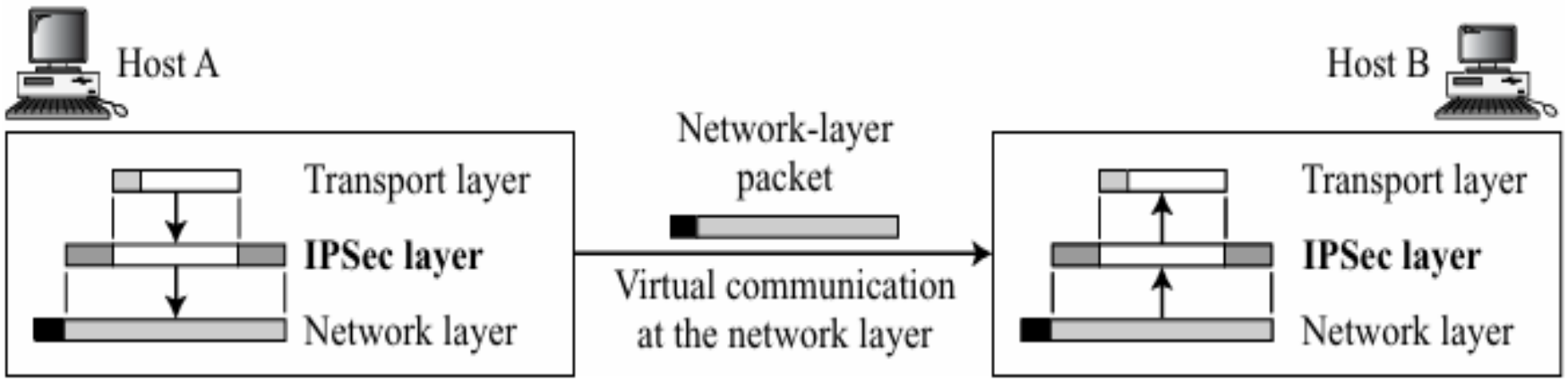
- In transport mode, IPSec protects what is delivered from the transport layer to the network layer. In other words, transport mode protects the network layer payload, the pay load to be encapsulated in the network layer.
- Transport mode is normally used when we need host-to-host (end-to-end) protection of data.
- The sending host uses IPSec to authenticate and/or encrypt the payload delivered from the transport layer.
- The receiving host uses IPSec to check the authentication and/or decrypt the IP packet and deliver it to the transport layer.

IPSec in transport mode



IPSec in transport mode does not protect the IP header; it only protects the information coming from the transport layer.

Transport mode in action



IPSec in transport mode does not protect the IP header; it only protects the information coming from the transport layer.

Transport mode is normally used when we need host-to-host (end-to-end) protection of data.

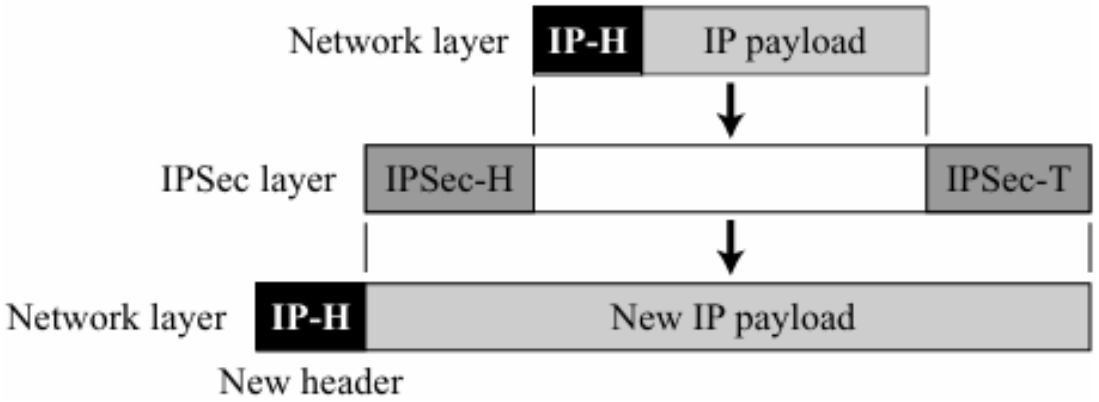
The sending host uses IPSec to authenticate and/or encrypt the payload delivered from the transport layer.

The receiving host uses IPSec to check the authentication and/or decrypt the IP packet and deliver it to the transport layer.

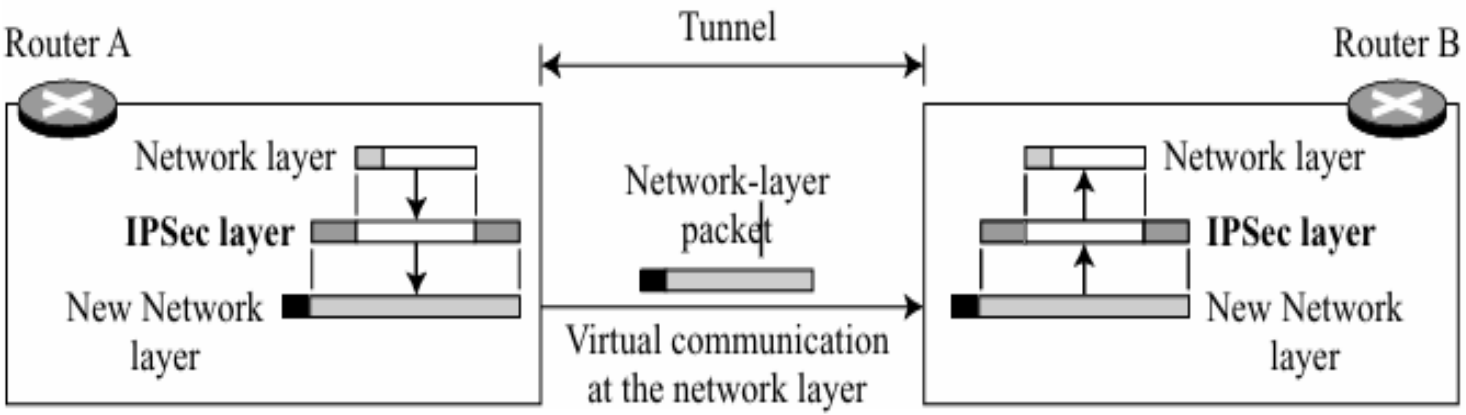
Tunnel Mode

In tunnel mode, IPSec protects the entire IP packet. It takes an IP packet, including the header, applies IPSec security methods to the entire packet, and then adds a new IP header.

IPSec in tunnel mode



Tunnel mode in action



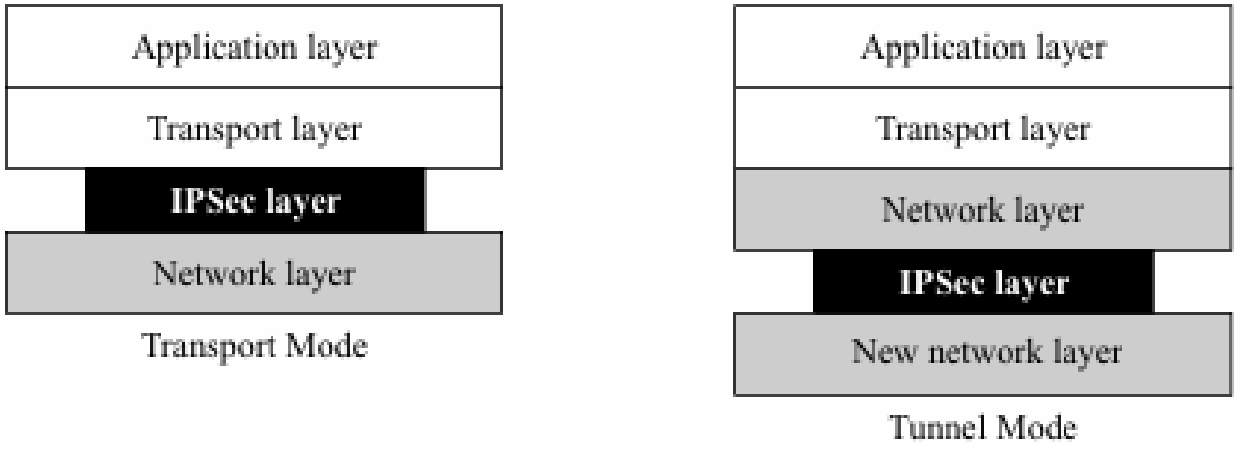
- Tunnel mode is used when either the sender or the receiver is not a host. The entire original packet is protected from intrusion between the sender and the receiver, as if the whole packet goes through an imaginary tunnel. IPSec in tunnel mode protects the original IP header.

IPSec in tunnel mode protects the original IP header.

Comparison

In transport mode, the IPSec layer comes between the transport layer and the network layer. In tunnel mode, the flow is from the network layer to the IPSec layer and then back to the network layer again.

Transport mode versus tunnel mode



IPSec defines two protocols

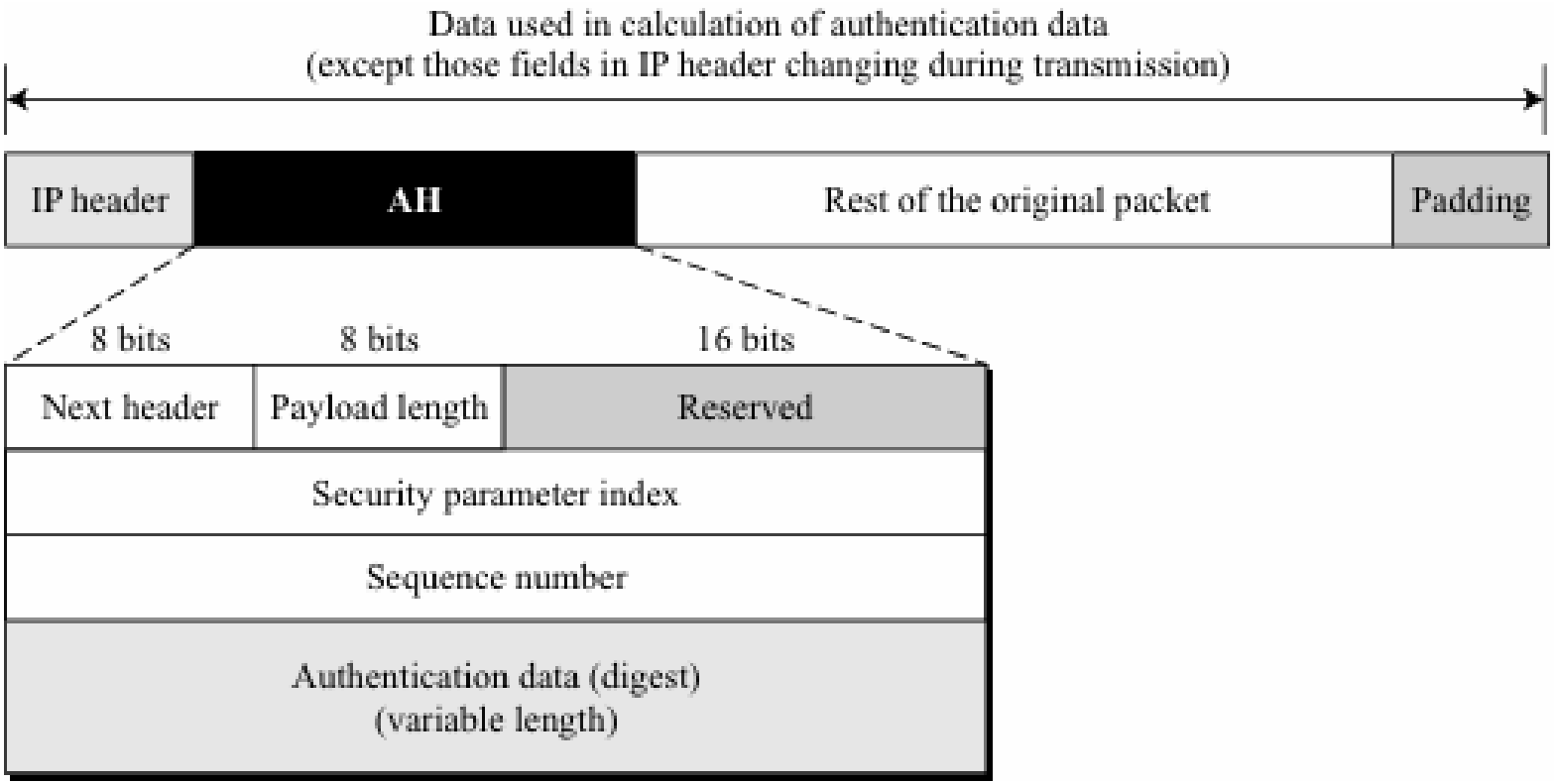
- Authentication Header (AH) Protocol
- Encapsulating Security Payload (ESP) Protocol

Purpose: Authentication and/or encryption for packets at the IP level.

Authentication Header (AH)

The Authentication Header (AH) Protocol is designed to authenticate the source host and to ensure the integrity of the payload carried in the IP packet. The protocol uses a hash function and a symmetric key to create a message digest; the digest is inserted in the authentication header. The AH is then placed in the appropriate location, based on the mode (transport or tunnel).

Authentication Header (AH) protocol



When an IP datagram carries an authentication header, the original value in the protocol field of the IP header is replaced by the value 51.

A field inside the authentication header (the next header field) holds the original value of the protocol field (the type of payload being carried by the IP datagram). The addition of an authentication header follows these steps:

- An authentication header is added to the payload with the authentication data field set to 0

- Padding may be added to make the total length even for a particular hashing algorithm.

- Hashing is based on the total packet. However, only those fields of the IP header that do not change during transmission are included in the calculation of the message digest (authentication data).
- The authentication data are inserted in the authentication header.
- The IP header is added after changing the value of the protocol field to 51.

Description of Field

- **Next header.** The 8-bit next header field defines the type of payload carried by the IP datagram (such as TCP, UDP, ICMP, or OSPF). It has the same function as the protocol field in the IP header before encapsulation. In other words, the process copies the value of the protocol field in the IP datagram to this field. The value of the protocol field in the new IP datagram is now set to 51 to show that the packet carries an authentication header.

- **Payload length.** The name of this 8-bit field is misleading. It does not define the length of the payload; it defines the length of the authentication header in 4-byte multiples, but it does not include the first 8 bytes.
- **Security parameter index.** The 32-bit security parameter index (SPI) field plays the role of a virtual circuit identifier and is the same for all packets sent during a connection called a Security Association (discussed later).
- **Sequence number.** A 32-bit sequence number provides ordering information for a sequence of datagrams. The sequence numbers prevent a playback. Note that the sequence number is not repeated even if a packet is retransmitted. A sequence number does not wrap around after it reaches 2^32; a new connection must be established.

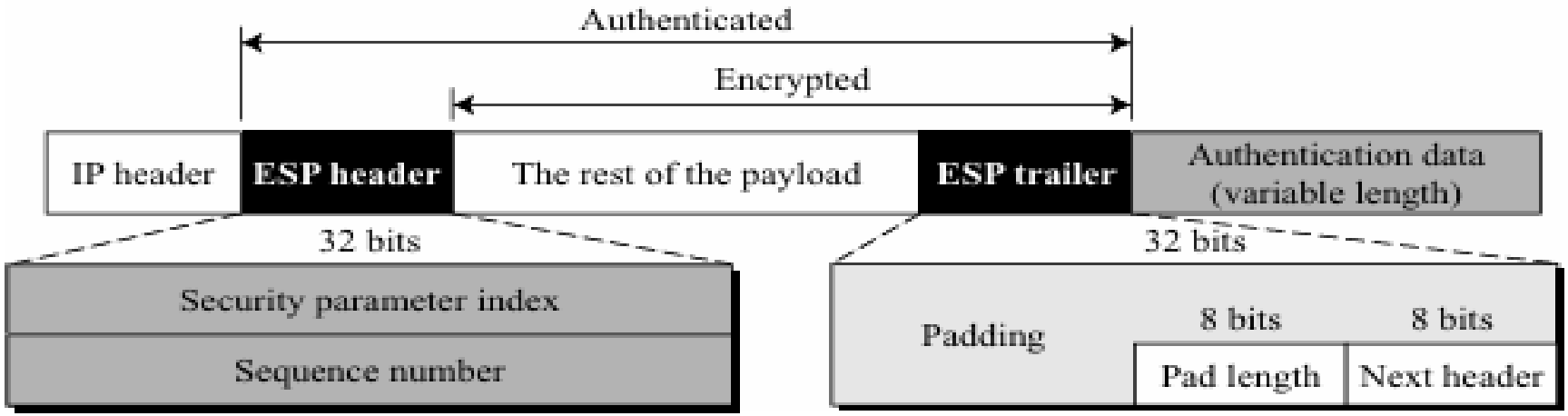
- **Authentication data.** Finally, the authentication data field is the result of applying a hash function to the entire IP datagram except for the fields that are changed during transit (e.g., time-to-live).

Note:-The AH protocol provides source authentication and data integrity, but not privacy.

Encapsulating Security Payload (ESP)

- The AH protocol does not provide privacy, only source authentication and data integrity.
- IPSec later defined an alternative protocol, Encapsulating Security Payload (ESP), that provides source authentication, integrity, and privacy.
- ESP adds a header and trailer. Note that ESP's authentication data are added at the end of the packet, which makes its calculation easier.

ESP


Authenticated
Encrypted
IP header
ESP header
The rest of the payload
ESP trailer
Authentication data
(variable length)
32 bits
Security parameter index
Sequence number
32 bits
Padding
8 bits
8 bits
Pad length
Next header

The ESP procedure

- When an IP datagram carries an ESP header and trailer, the value of the protocol field in the IP header is 50.

- A field inside the ESP trailer (the next-header field) holds the original value of the protocol field (the type of payload being carried by the IP data gram, such as TCP or UDP).

- The ESP procedure follows these steps:
  - An ESP trailer is added to the payload.
  - The payload and the trailer are encrypted.
  - The ESP header is added.
  - The ESP header, payload, and ESP trailer are used to create the authentication data
  - The authentication data are added to the end of the ESP trailer.
  - The IP header is added after changing the protocol value to 50.

ESP Field (Description)

- **Security parameter index.** The 32-bit security parameter index field is similar to that defined for the AH protocol.
- **Sequence number.** The 32-bit sequence number field is similar to that defined for the AH protocol.
- **Padding.** This variable-length field (0 to 255 bytes) of 0s serves as padding.
- **Pad length.** The 8-bit pad-length field defines the number of padding bytes. The value is between 0 and 255; the maximum value is rare.
- **Next header.** The 8-bit next-header field is similar to that defined in the AH protocol. It serves the same purpose as the protocol field in the IP header before encapsulation.

- Authentication data. Finally, the authentication data field is the result of applying an authentication scheme to parts of the datagram. Note the difference between the authentication data in AH and ESP. In AH, part of the IP header is included in the calculation of the authentication data; in ESP, it is not.

Note:-ESP provides source authentication, data integrity, and privacy.

Services Provided by IPSec

Services	*AH*	*ESP*
Access control	yes	yes
Message authentication (message integrity)	yes	yes
Entity authentication (data source authentication)	yes	yes
Confidentiality	**no**	yes
Replay attack protection	yes	yes

- **Access Control**

IPSec provides access control indirectly using a Security Association Database (SAD). When a packet arrives at a destination, and there is no Security Association already established for this packet, the packet is discarded.

- **Message Integrity**

Message integrity is preserved in both AH and ESP. A digest of data is created and sent by the sender to be checked by the receiver.

- **Entity Authentication**

The Security Association and the keyed-hash digest of the data sent by the sender authenticate the sender of the data in both AH and ESP.

- **Confidentiality**

The encryption of the message in ESP provides confidentiality. AH, however, does not provide confidentiality. If confidentiality is needed, one should use ESP instead of AH.

❑ Replay Attack Protection

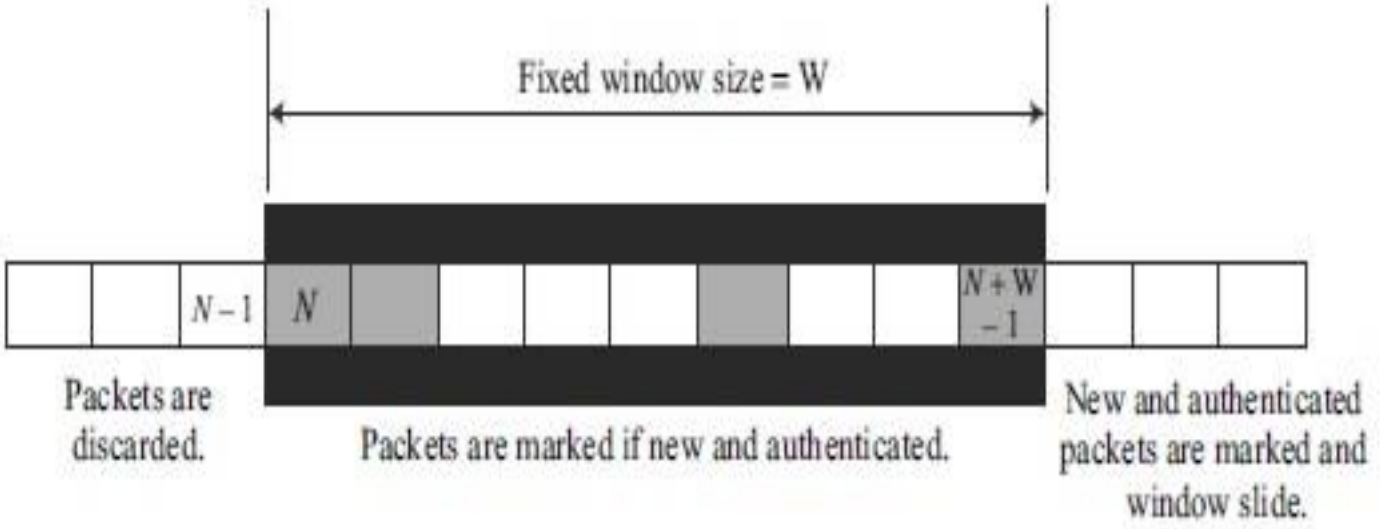
In both protocols, the replay attack is prevented by using sequence numbers and a sliding receiver window. Each IPSec header contains a unique sequence number when the Security Association is established. The number starts from 0 and increases until the value reaches $2^{32} - 1$.

When the sequence number reaches the maximum, it is reset to 0 and, at the same time, the old Security Association (see the next section) is deleted and a new one is established.

To prevent processing duplicate packets, IPSec mandates the use of a fixed-size window at the receiver. The size of the window is determined by the receiver with a default value of 64.

The window is of a fixed size, W. The shaded packets signify received packets that have been checked and authenticated.

Replay Window



When a packet arrives at the receiver, one of three things can happen, depending on the value of the sequence number.

1. The sequence number of the packet is less than N. This puts the packet to the left of the window. In this case, the packet is discarded. It is either a duplicate or its arrival time has expired.

2. The sequence number of the packet is between N and (N + W − 1), inclusive. This puts the packet inside the window. In this case, if the packet is new (not marked) and it passes the authentication test, the sequence number is marked and the packet is accepted. Otherwise, it is discarded.

3. The sequence number of the packet is greater than (N + W − 1). This puts the packet to the right of the window. In this case, if the packet is authenticated, the corresponding sequence number is marked and the window slides to the right to cover the newly marked sequence number. Otherwise, the packet is discarded.

Note that it may happen that a packet arrives with a sequence number much larger than (N + W) (very far from the right edge of the window). In this case, the sliding of the window may cause many unmarked numbers to fall to the left of the window. These packets, when they arrive, will never be accepted; their time has expired.

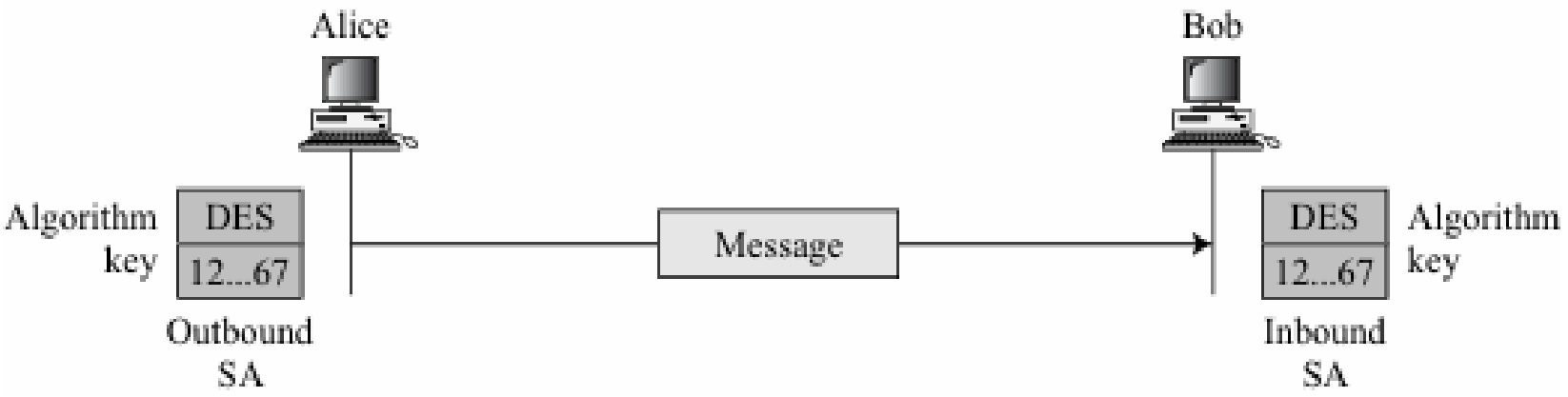
For example, if a packet arrives with sequence number (N + W + 3), the window slides and the left edge will be at the beginning of (N + 3). This means the sequence number (N + 2) is now out of the window. If a packet arrives with this sequence number, it will be discarded.

SECURITY ASSOCIATION (SA)

- A Security Association is a contract between two parties , it creates a secure channel between them.
- Let us assume that Alice needs to unidirectionally communicate with Bob. If Alice and Bob are interested only in the confidentiality aspect of security, they can get a shared secret key between themselves. We can say that there are two Security Associations (SAs) between Alice and Bob; one outbound SA and one inbound SA.

- Each of them stores the value of the key in a variable and the name of the encryption/ decryption algorithm in another. Alice uses the algorithm and the key to encrypt a message to Bob; Bob uses the algorithm and the key when he needs to decrypt the message received from Alice.

Simple SA


Alice
Bob
Algorithm
key
DES
12...67
Outbound
SA
Message
DES
12...67
Algorithm
key
Inbound
SA

Security Association Database (SAD)

A Security Association can be very complex.

This is particularly true if Alice wants to send messages to many people and Bob needs to receive messages from many people.

In addition, each site needs to have both inbound and outbound SAs to allow bidirectional communication. In other words, we need a set of SAs that can be collected into a database.

This database is called the Security Association Database (SAD). The database can be thought of as a two-dimensional table with each row defining a single SA. Normally, there are two SADs, one inbound and one outbound.

SAD

< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							

Security Association Database

Legend:

SPI: Security Parameter Index	SN: Sequence Number
DA: Destination Address	OF: Overflow Flag
AH/ESP: Information for either one	ARW: Anti-Replay Window
P: Protocol	LT: Lifetime
Mode: IPSec Mode Flag	MTU: Path MTU (Maximum Transfer Unit)

When a host needs to send a packet that must carry an IPSec header, the host needs to find the corresponding entry in the outbound SAD to find the information for applying security to the packet.

Similarly, when a host receives a packet that carries an IPSec header, the host needs to find the corresponding entry in the inbound SAD to find the information for checking the security of the packet.

This searching must be specific in the sense that the receiving host needs to be sure that correct information is used for processing the packet. Each entry in an inbound SAD is selected using a triple index: security parameter index, destination address, and protocol.

- **Security Parameter Index**.

  The security parameter index (SPI) is a 32-bit number that defines the SA at the destination. As we will see later, the SPI is determined during the SA negotiation. The same SPI is included in all IPSec packets belonging to the same inbound SA.

- **Destination Address**.

  The second index is the destination address of the host. We need to remember that a host in the Internet normally has one unicast destination address, but it may have several multicast addresses. IPSec requires that the SAs be unique for each destination address.

- **Protocol**.

  IPSec has two different security protocols: AH and ESP. To separate the parameters and information used for each protocol, IPSec requires that a destination define a different SA for each protocol.

Typical SA Parameters

Parameters	*Descriptions*
Sequence Number Counter	This is a 32-bit value that is used to generate sequence numbers for the AH or ESP header.
Sequence Number Overflow	This is a flag that defines a station's options in the event of a sequence number overflow.
Anti-Replay Window	This detects an inbound replayed AH or ESP packet.
AH Information	This section contains information for the AH protocol: 1. Authentication algorithm 2. Keys 3. Key lifetime 4. Other related parameters
ESP Information	This section contains information for the ESP protocol: 1. Encryption algorithm 2. Authentication algorithm 3. Keys 4. Key lifetime 5. Initiator vectors 6. Other related parameters
SA Lifetime	This defines the lifetime for the SA.
IPSec Mode	This defines the mode, transport or tunnel.
Path MTU	This defines the path MTU (fragmentation).

Security Policy & Database

Another import aspect of IPSec is the Security Policy (SP), which defines the type of security applied to a packet when it is to be sent or when it has arrived. Before using the SAD, a host must determine the predefined policy for the packet.

- Security Policy Database

- Each host that is using the IPSec protocol needs to keep a Security Policy Database (SPD). Again, there is a need for an inbound SPD and an outbound SPD.
- Each entry in the SPD can be accessed using a six-tuple index

Index	Policy
<SA, DA, Name, P, SPort, DPort>	
<SA, DA, Name, P, SPort, DPort>	
<SA, DA, Name, P, SPort, DPort>	
<SA, DA, Name, P, SPort, DPort>	

Legend:

SA: Source Address
SPort: Source Port
DA: Destination Address
DPort: Destination Port
P: Protocol

- Source and destination addresses can be unicast, multicast, or wildcard addresses.
- The name usually defines a DNS entity.
- The protocol is either AH or ESP.
- The source and destination ports are the port addresses for the process running at the source and destination hosts.

Outbound SPD

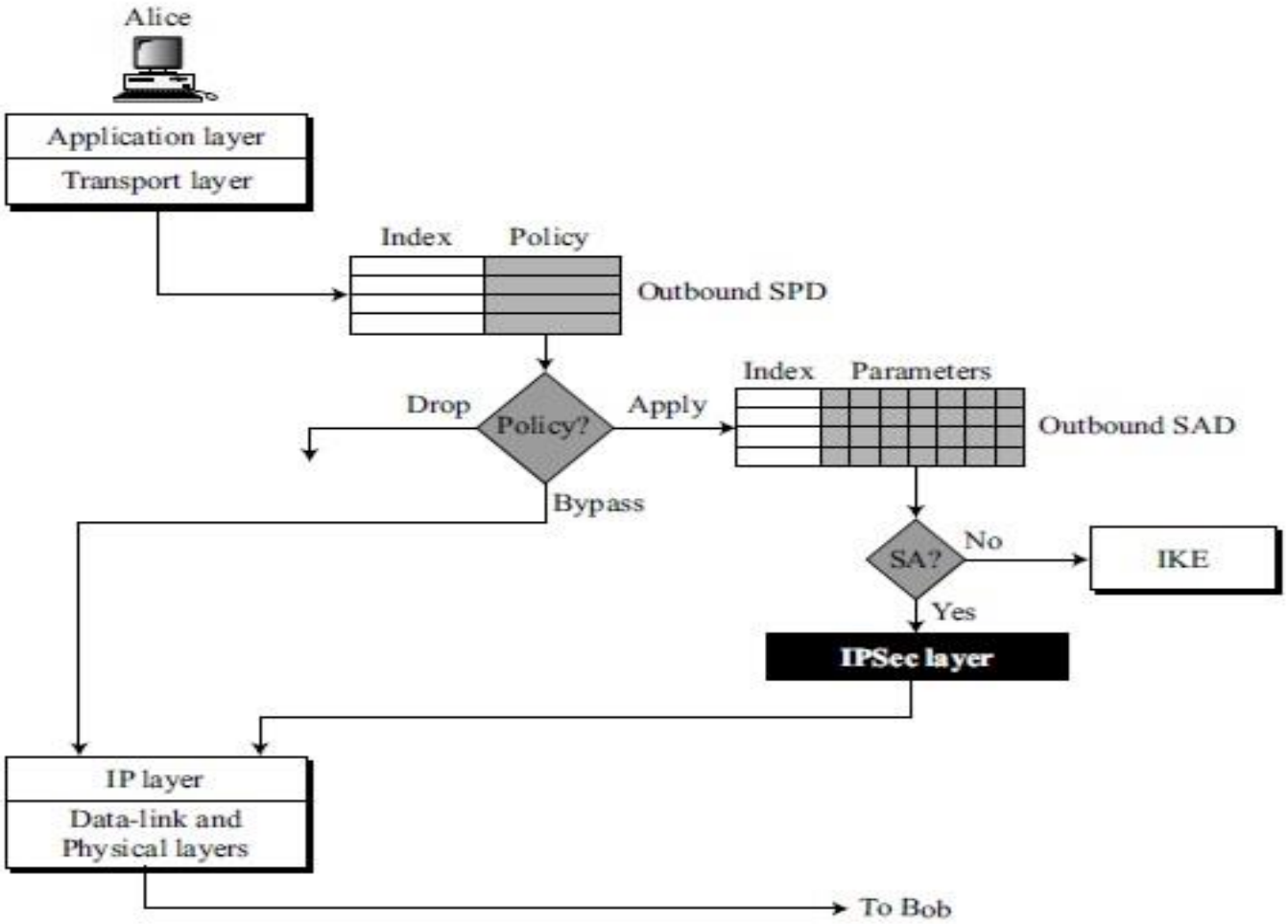
When a packet is to be sent out, the outbound SPD is consulted.
The input to the outbound SPD is the six-tuple index;
The output is one of the three following cases:

- Drop : This means that the packet defined by the index cannot be sent; it is dropped.
- Bypass: This means that there is no policy for the packet with this policy index; the packet is sent, bypassing the security header application.
- Apply : In this case, the security header is applied. Two situations may occur.

(a) If an outbound SA is already established, the triple SA index is returned that selects the corresponding SA from the outbound SAD. The AH or ESP header is formed; encryption, authentication, or both are applied based on the SA selected. The packet is transmitted.

(b) If an outbound SA is not established yet, the Internet Key Exchange (IKE) protocol is called to create an outbound and inbound SA for this traffic. The outbound SA is added to the outbound SAD by the source; the inbound SA is added to the inbound SAD by the destination.

Outbound Processing



Inbound SPD

The input to the inbound SPD is the six-tuple index; the output is one of the three following cases:
Discard. This means that the packet defined by that policy must be dropped.

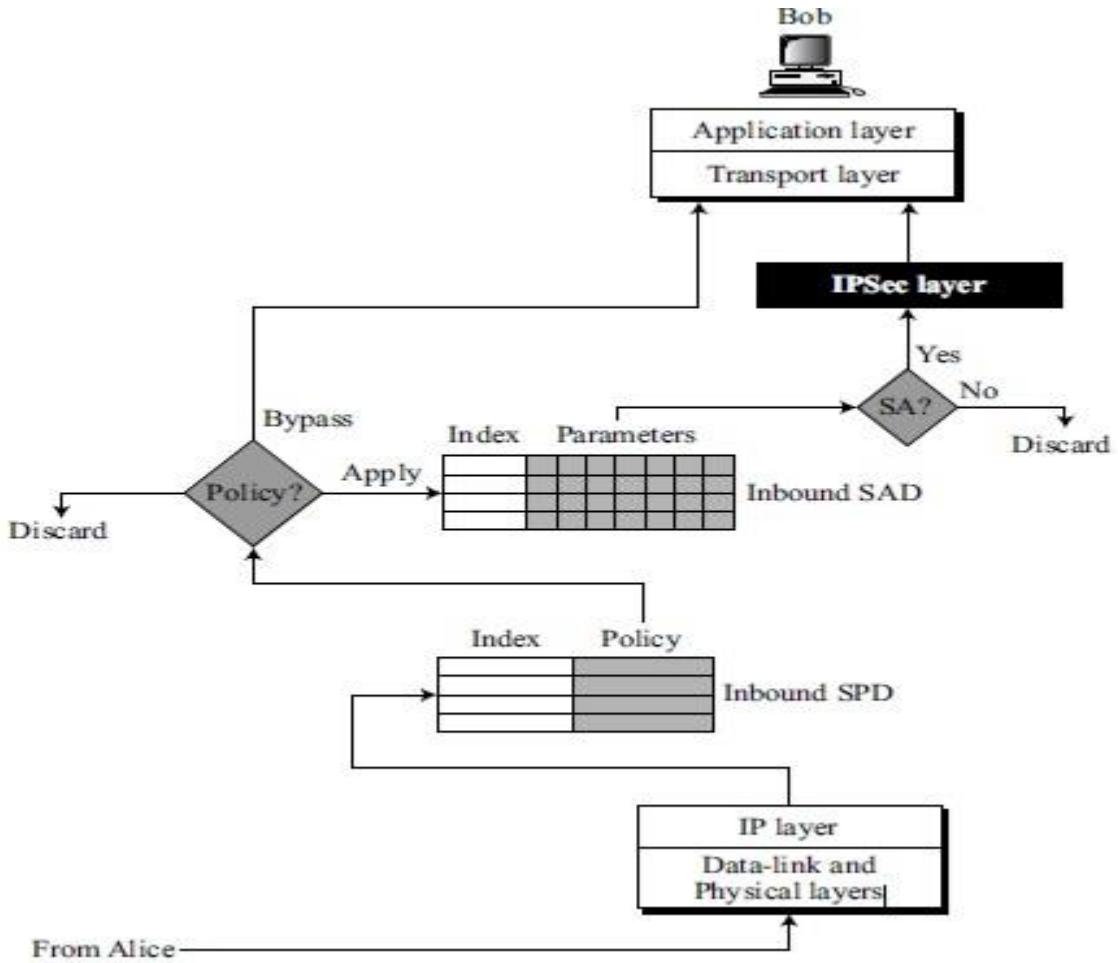
Bypass. This means that there is no policy for a packet with this policy index; the packet is processed, ignoring the information from AH or ESP header. The packet is delivered to the transport layer.

Apply. In this case, the security header must be processed. Two cases may occur:

(a) If an inbound SA is already established, the triple SA index is returned that selects the corresponding inbound SA from the inbound SAD. Decryption, authentication, or both are applied. If the packet passes the security criteria, the AH or ESP header is discarded and the packet is delivered to the transport layer.

(b) If an SA is not yet established, the packet must be discarded.

Inbound Processing

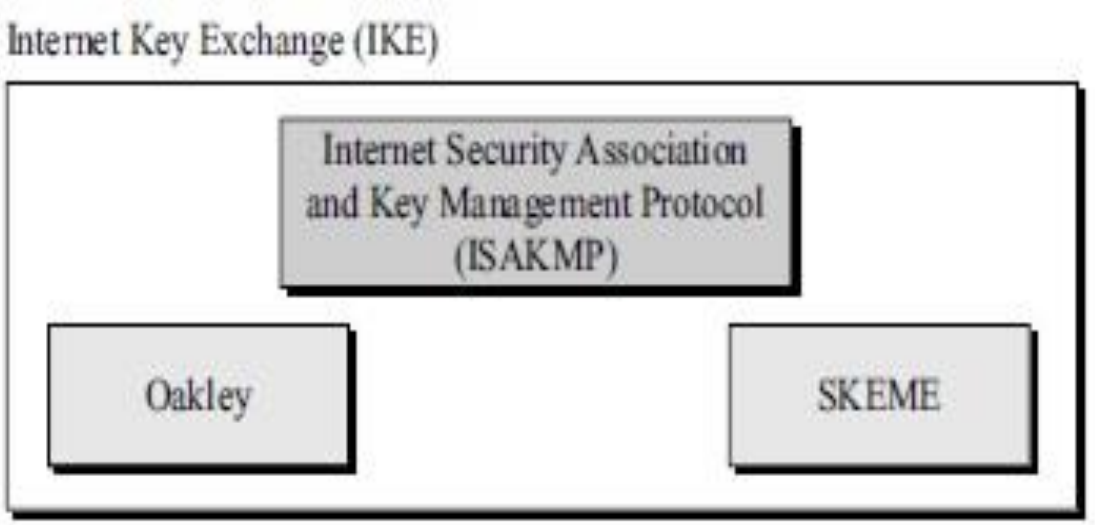


Internet Key Exchange Protocol

- The Internet Key Exchange (IKE) is a protocol designed to create both inbound and outbound Security Associations.
- When a peer needs to send an IP packet, it consults the Security Policy Database (SPDB) to see if there is an SA for that type of traffic. If there is no SA, IKE is called to establish one.

Note : IKE creates SAs for IPSec.

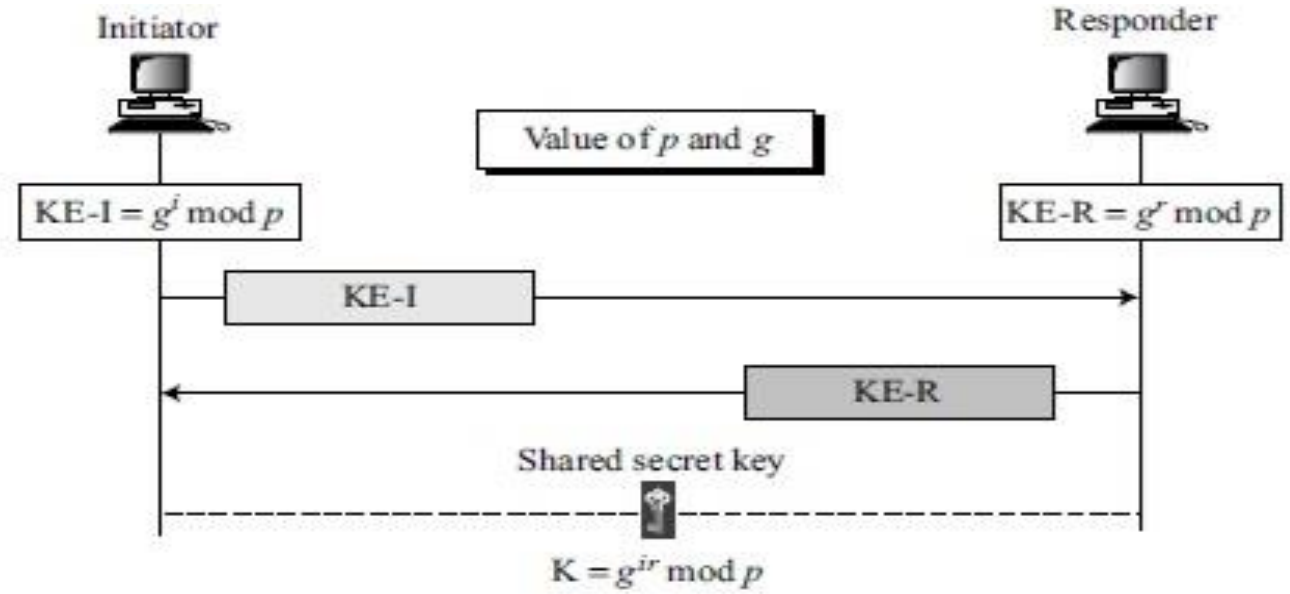
IKE Component



IKE is a complex protocol based on three other protocols: Oakley, SKEME, and ISAKMP

- The Oakley protocol was developed by Hilarie Orman. It is a key creation protocol based on the Diffie-Hellman key-exchange method, but with some improvements. Oakley is a free-formatted protocol in the sense that it does not define the format of the message to be exchanged. (IKE uses its ideas).
- SKEME, designed by Hugo Krawcyzk, is another protocol for key exchange. It uses public-key encryption for entity authentication in a key-exchange protocol. We will see shortly that one of the methods used by IKE is based on SKEME.
- The Internet Security Association and Key Management Protocol (ISAKMP) is a protocol designed by the National Security Agency (NSA) that actually implements the exchanges defined in IKE. It defines several packets, protocols, and parameters that allow the IKE exchanges to take place in standardized, formatted messages to create SAs. The ISAKMP is the carrier protocol that implements IKE.

Improved DH Key Exchange

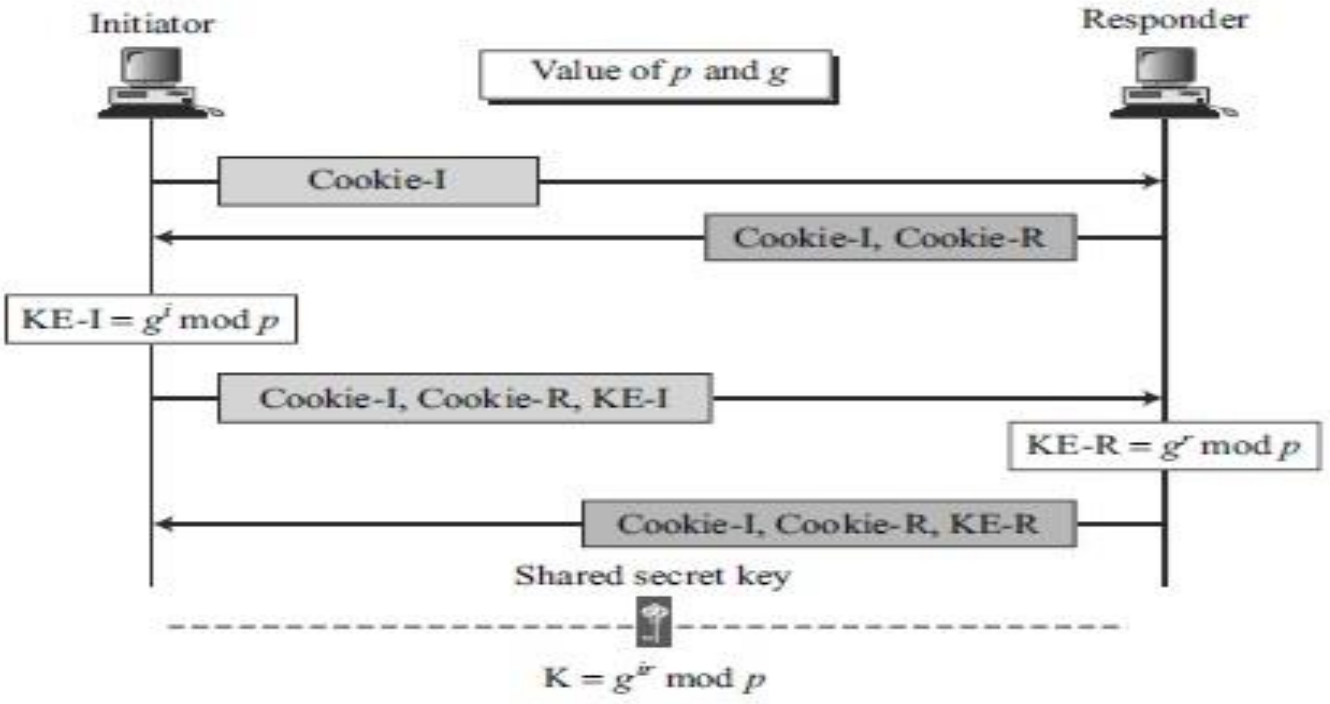


The Diffie-Hellman protocol has some weaknesses that need to be eliminated before it is suitable as an Internet key exchange.

- Clogging Attack

  - The first issue with the Diffie-Hellman protocol is the clogging attack or denial-of service attack. A malicious intruder can send many half-key (g^X mod q) messages to Bob, pretending that they are from different sources. Bob then needs to calculate different responses (g^Y mod q) and at the same time calculate the full-key (g^{XY} mod q).
  - This keeps Bob so busy that he may stop responding to any other messages. He denies services to clients.This can happen because the Diffie-Hellman protocol is computationally intensive

DH With Cookies



- The cookie is the result of hashing a unique identifier of the peer (such as IP address, port number, and protocol), a secret random number known to the party that generates the cookie, and a timestamp.

  - The initiator sends its own cookie; the responder its own. Both cookies are repeated, unchanged, in every following message.
  - The calculations of half-keys and the session key are postponed until the cookies are returned.
  - If any of the peers is a hacker attempting a clogging attack, the cookies are not returned; the corresponding party does not spend the time and effort to calculate the half-key or the session key. For example, if the initiator is a hacker using a bogus IP address, the initiator does not receive the second message and cannot send the third message. The process is aborted.

❑ Replay Attack

- Diffie-Hellman is vulnerable to a replay attack i.e. the information from one session can be replayed in a future session by a malicious intruder.

- To prevent this, we can add nonces to the third and fourthmessages to preserve the freshness of the message.

- Man-In-The-Middle Attack

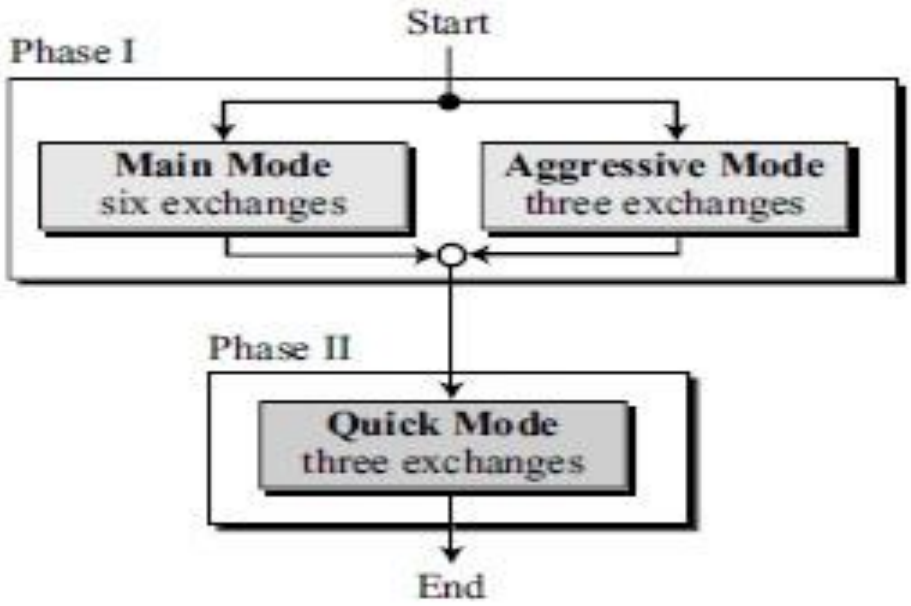
  - Authentication of the messages exchanged (message integrity) and the authentication of the parties involved (entity authentication) require that each party proves his/her claimed identity. To do this, each must prove that it possesses a secret.

  - In IKE, the secret can be one of the following:

(a) A preshared secret key
(b)A preknown encryption/decryption public-key pair. An entity must show that a message encrypted with the announced public key can be decrypted with the corresponding private key.
(c) A preknown digital signature public-key pair. An entity must show that it can sign a message with its private key which can be verified with its announced public key

IKE Phases

IKE is divided into two phases: phase I and phase II. Phase I creates SAs for phase II;phase II creates SAs for a data exchange protocol such as IPSec.

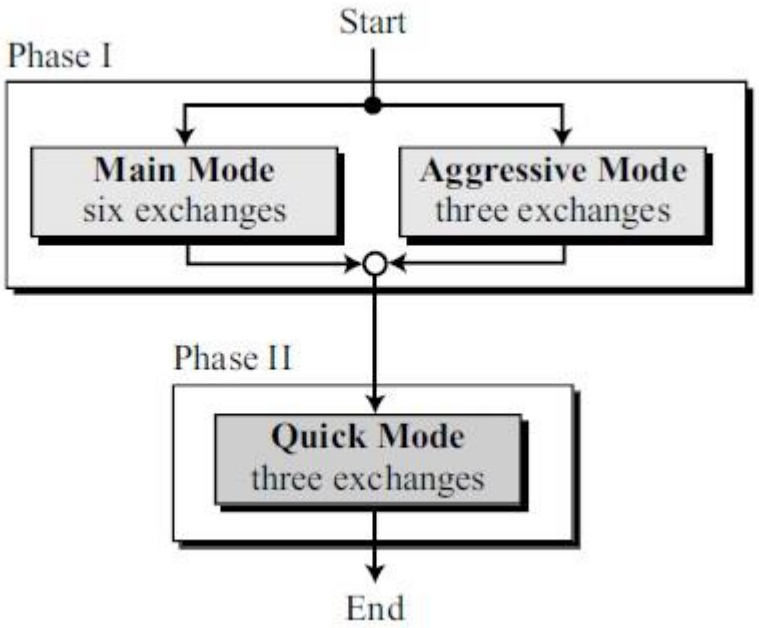


IKE is divided into two phases:

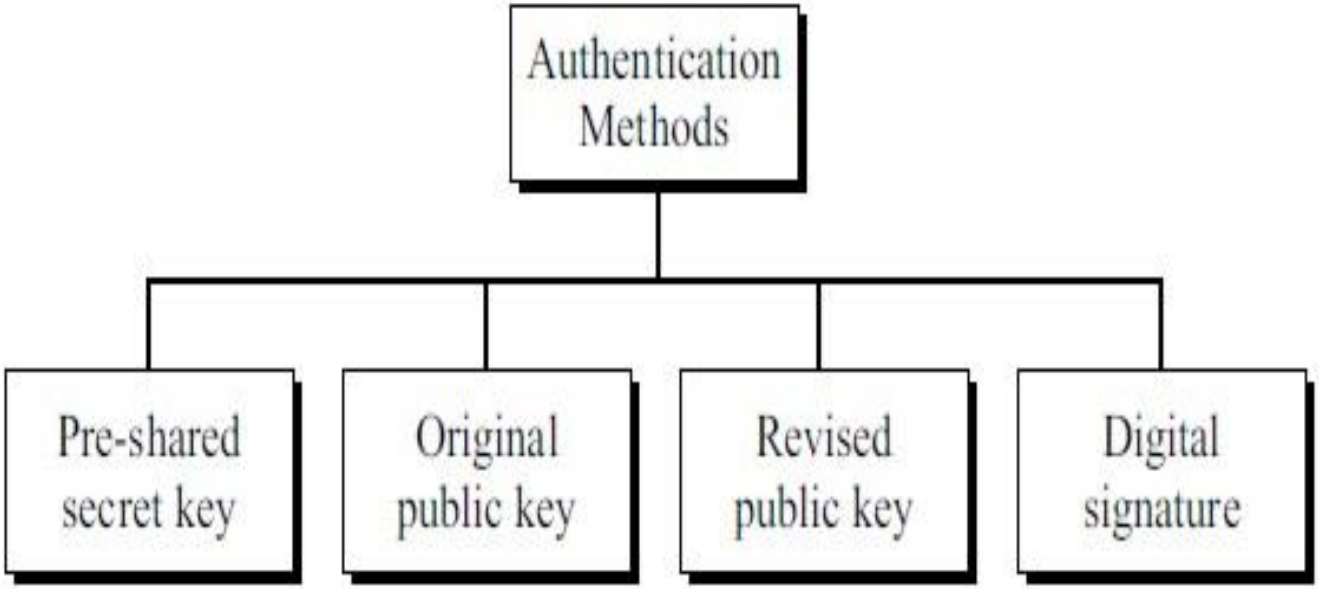
- Phase I creates SAs for phase II;
- Phase II creates SAs for a data exchange protocol such as IPSec.

Modes

To allow for a variety of exchange methods, IKE has defined modes for the phases.



Main Mode


Authentication Methods
Pre-shared secret key
Original public key
Revised public key
Digital signature

- In the main mode, the initiator and the responder exchange six messages.
- In the first two messages, they exchange cookies (to protect against a clogging attack) and negotiate the SA parameters. The initiator sends a series of proposals; the responder selects one of them.
- When the first two messages are exchanged, the initiator and the responder know the SA parameters and are confident that the other party exists (no clogging attack occurs).
- In the third and fourth messages, the initiator and responder usually exchange their half-keys (gi and gr of the Diffie-Hellman method) and their nonces (for replay protection).

Note: The half-keys and nonces are not sent with the first two messages because the two parties must first ensure that a clogging attack is not possible.

- After exchanging the third and fourth messages, each party can calculate the common secret between them in addition to its individual hash digest. The common secret SKEYID (secret key ID) is dependent on kind of method .

$$SKEYID = prf(\text{preshared-key}, N\text{-}I \mid N\text{-}R) \qquad \text{(preshared-key method)}$$

$$SKEYID = prf(N\text{-}I \mid N\text{-}R, g^{ir}) \qquad \text{(public-key method)}$$

$$SKEYID = prf(\text{hash}(N\text{-}I \mid N\text{-}R), Cookie\text{-}I \mid Cookie\text{-}R) \qquad \text{(digital signature)}$$

Other common secrets are calculated as follows:

$$SKEYID\_d = prf(SKEYID, g^{ir} \mid Cookie\text{-}I \mid Cookie\text{-}R \mid 0)$$

$$SKEYID\_a = prf(SKEYID, SKEYID\_d \mid g^{ir} \mid Cookie\text{-}I \mid Cookie\text{-}R \mid 1)$$

$$SKEYID\_e = prf(SKEYID, SKEYID\_a \mid g^{ir} \mid Cookie\text{-}I \mid Cookie\text{-}R \mid 2)$$

- SKEYID_d (derived key) is a key to create other keys.
- SKEYID_a is the authentication key used, during the negotiation phase
- SKEYID_e is used for the encryption key, used during the negotiation phase

Note : The key for Prf (Pseudo random function) is a keyed-hash function and defined during the negotiation phase (always is SKEYID).

The two parties also calculate two hash digests, HASH-I and HASH-R, which are used in three of the four methods in the main mode.

$$\text{HASH-I} = \textit{prf}(\text{SKEYID}, \text{KE-I} \mid \text{KE-R} \mid \text{Cookie-I} \mid \text{Cookie-R} \mid \text{SA-I} \mid \text{ID-I})$$

$$\text{HASH-R} = \textit{prf}(\text{SKEYID}, \text{KE-I} \mid \text{KE-R} \mid \text{Cookie-I} \mid \text{Cookie-R} \mid \text{SA-I} \mid \text{ID-R})$$

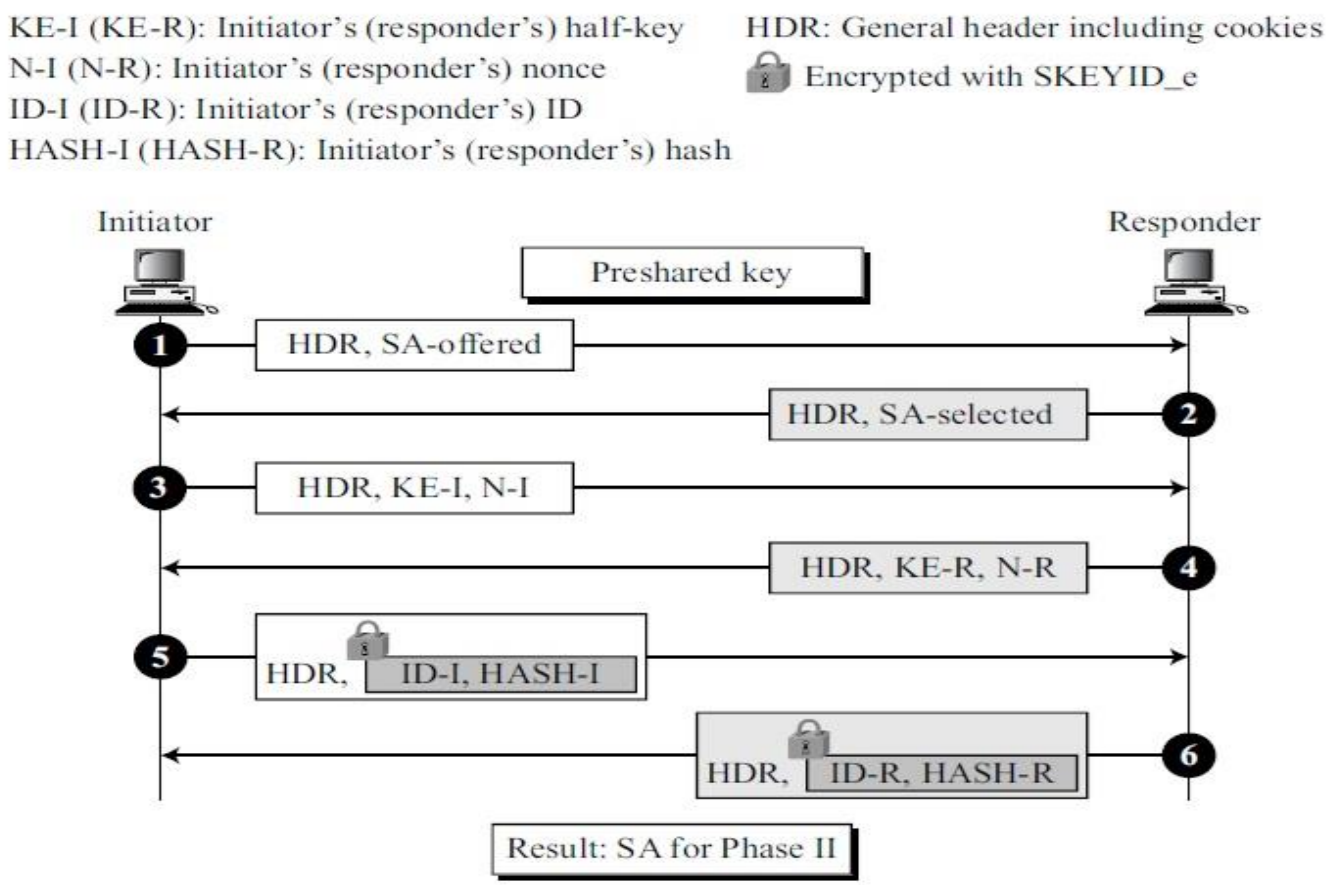
- The first digest uses ID-I, while the second uses ID-R.
- Both use SA-I, the entire SA data sent by the initiator.
- None of them include the proposal selected by the responder.

- The idea is to protect the proposal sent by the initiator by preventing an intruder from making changes. For example, an intruder might try to send a list of proposals more vulnerable to attack.
- Similarly, if the SA is not included, an intruder might change the selected proposal to one more favorable to himself.
- Party does not need to know the ID of the other party in the calculation of the HASHs.

- After calculating the keys and hashes, each party sends the hash to the other party to authenticate itself. The initiator sends HASH-I to the responder as proof that she is Alice.
- Only Alice knows the authentication secret and only she can calculate HASH-I. If the HASH-I then calculated by Bob matches the HASH-I sent by Alice, she is authenticated.

Note: When Bob calculates HASH-I, he needs Alice's ID and vice versa. In some methods, the ID is sent by previous messages; in others it is sent with the hash, with both the hash and the ID encrypted by SKEYID_e.

Pre-shared Secret-Key Method



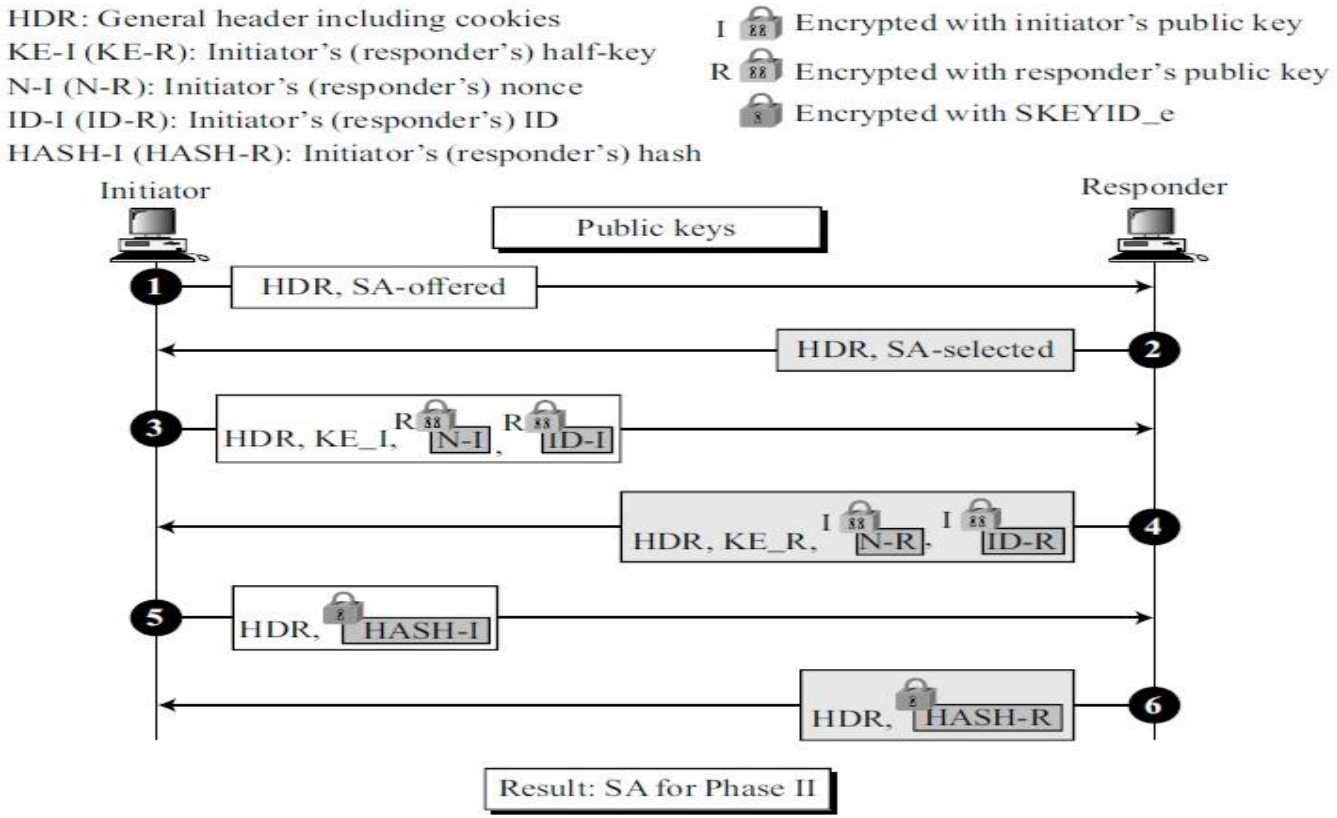
In the preshared secret-key method, a symmetric key is used for authentication of the peers to each other.

- In the first two messages, the initiator and responder exchange cookies (inside the general header) and SA parameters.
- In the next two messages, they exchange the halfkeys and the nonces .
- Now the two parties can create SKEYID and the two keyed hashes (HASH-I and HASH-R).
- In the fifth and sixth messages, the two parties exchange the created hashes and their IDs. To protect the IDs and hashes, the last two messages are encrypted with SKEYID_e.

The pre-shared key is the secret between Alice (initiator) and Bob (responder). Eve (intruder) does not have access to this key. Eve cannot create SKEYID and therefore cannot create either HASH-I or HASH-R.

Note : The IDs need to be exchanged in messages 5 and 6 to allow the calculation of the hash.

Original Public Key Method



In the original public-key method, the initiator and the responder prove their identities by showing that they possess a private key related to their announced public key

- The first two messages are the same as in the previous method. In the third message, the initiator sends its half-key, the nonce, and the ID. In the fourth message, the responder does likewise. However, the nonces and IDs are encrypted by the public key of the receiver and decrypted by the private key of the receiver

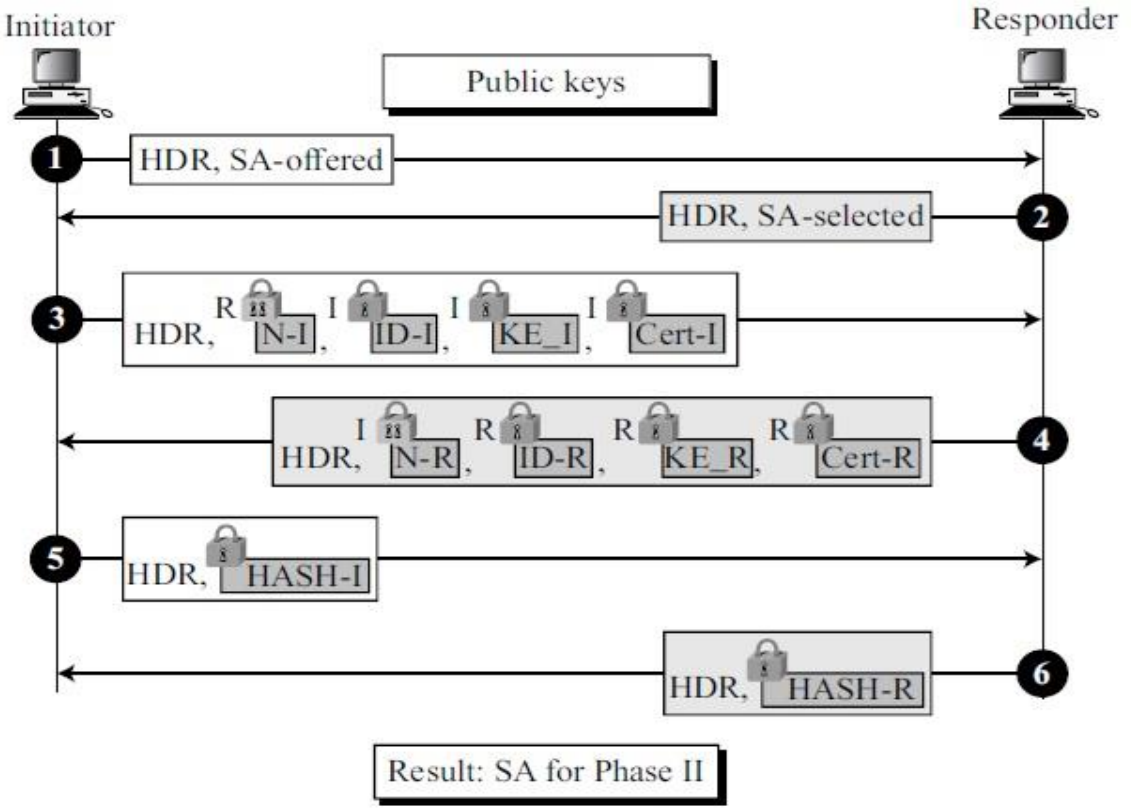
- One difference between this method and the previous one is that the IDs are exchanged with the third and fourth messages instead of the fifth and sixth messages.The fifth and sixth messages just carry the HASHs

Revised Public Key Method

- The original public-key method has some drawbacks.
  - First, two instances of public-key encryption/decryption place a heavy load on the initiator and responder.
  - Second, the initiator cannot send its certificate encrypted by the public key of the responder, since anyone could do this with a false certificate.
  - The method was revised so that the public key is used only to create a temporary secret key

HDR: General header including cookies
KE-I (KE-R): Initiator's (responder's) half-key
Cert-I (Cert-R): Initiator's (responder's) certificate
N-I (N-R): Initiator's (responder's) nonce
ID-I (ID-R): Initiator's (responder's) ID
HASH-I (HASH-R): Initiator's (responder's) hash

I Encrypted with initiator's public key
R Encrypted with responder's public key
R Encrypted with responder's secret key
I Encrypted with initiator's secret key
Encrypted with SKEYID_e

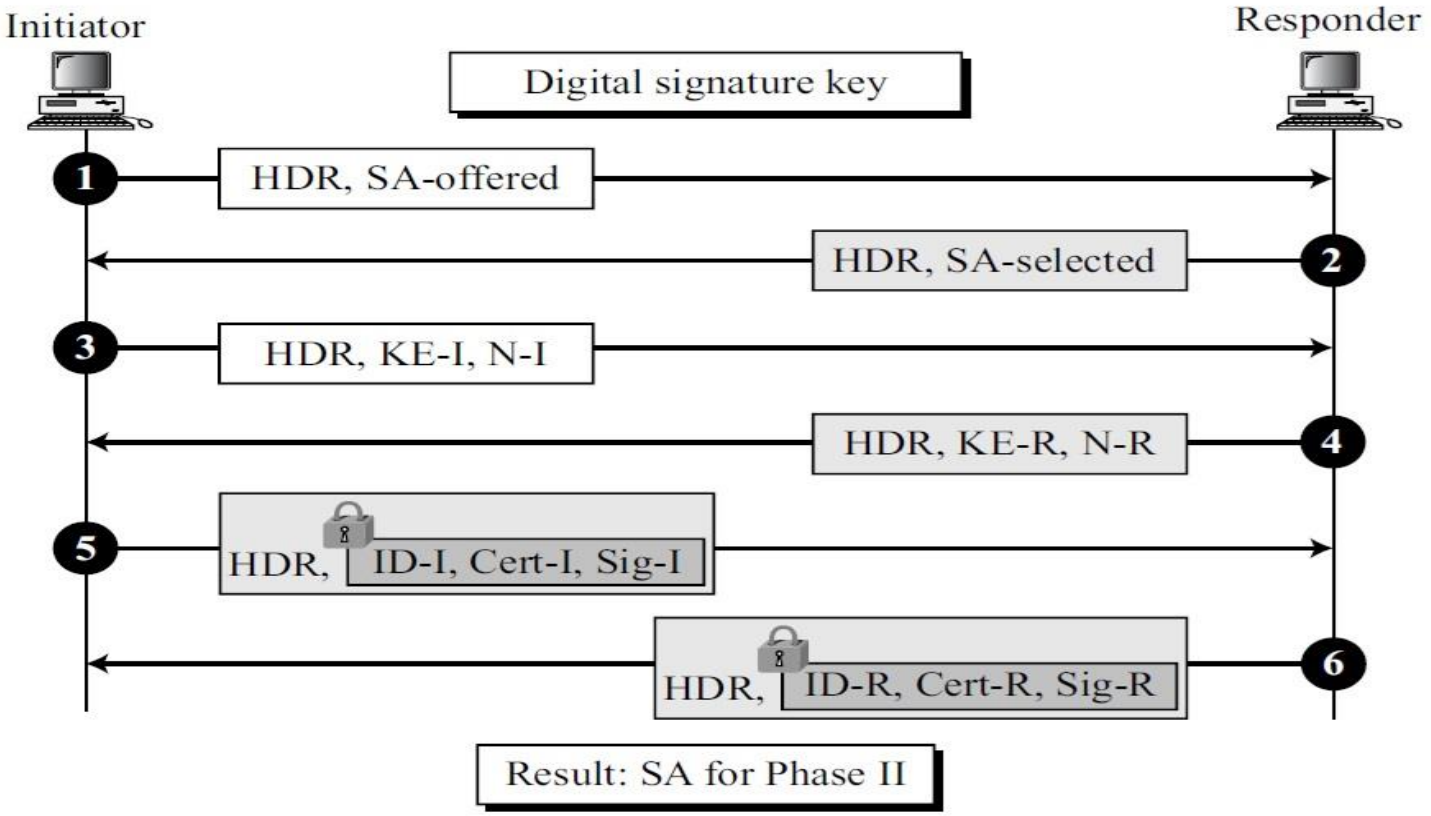


- Two temporary secret keys are created from a hash of nonces and cookies.
- The initiator uses the public key of the responder to send its nonce. The responder decrypts the nonce and calculates the initiator's temporary secret key. After that the half-key, the ID, and the optional certificate can be decrypted. The two temporary secret keys, K-I and K-R, are calculated as

$$K\text{-}I = prf(N\text{-}I, Cookie\text{-}I) \qquad K\text{-}R = prf(N\text{-}R, Cookie\text{-}R)$$

Digital Signature Method

- Each party shows that it possesses the certified private key related to a digital signature.



HDR: General header including cookies

Sig-I: Initiator's signature on messages 1–4

Sig-R: Initiator's signature on messages 1–5

Cert-I (Cert-R): Initiator's (responder's) certificate

N-I (N-R): Initiator's (responder's) nonce

KE-I (KE-R): Initiator's (responder's) half-key

ID-I (ID-R): Initiator's (responder's) ID

 Encrypted with SKEYID_e

- Note that in this method the sending of the certificates is optional. The certificate can be sent here because it can be encrypted with SKEYID_e, which does not depend on the signature key.

- Message 5, the initiator signs all the information exchanged in messages 1 to 4 with its signature key. The responder verifies the signature using the public key of the initiator, which authenticates the initiator.
- Message 6, the responder signs all the information exchanged with its signature key. The initiator verifies the signature

Aggressive Mode

Each aggressive mode is a compressed version of the corresponding main mode.

Instead of six messages, only three are exchanged. Messages 1 and 3 are combined to make the first message. Messages 2, 4, and 6 are combined to make the second message. Message 5 is sent as the third message. The idea is the same

Phase-II Quick Mode

The quick mode uses IKE SAs to create IPSec SAs (or SAs for any other protocol).